06-1-00

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b)

05/30/00 JC944 U.S. PTO

Attorney Docket No. 002950.P055

First Named Inventor or Application Identifier Gary Barnett et al.

Express Mail Label No. EL580090156US

JC784 U.S. PTO 09/583311 05/30/00

ADDRESS TO: **Assistant Commissioner for Patents**
Box Patent Application
Washington, D. C. 20231

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents.

1. _X_ Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)

2. _X_ Specification (Total Pages 25)
(preferred arrangement set forth below)
- Descriptive Title of the Invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to Microfiche Appendix
- Background of the Invention
- Brief Summary of the Invention
- Brief Description of the Drawings (if filed)
- Detailed Description
- Claims
- Abstract of the Disclosure

3. _X_ Drawings(s) (35 USC 113) (Total Sheets 15)

4. _X_ Oath or Declaration (Total Pages 5)
   a. _X_ Newly Executed (Original or Copy)
   b. ___ Copy from a Prior Application (37 CFR 1.63(d))
   (for Continuation/Divisional with Box 17 completed) **(Note Box 5 below)**
   i. ___ DELETIONS OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).

5. ____ Incorporation By Reference (useable if Box 4b is checked)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. ____ Microfiche Computer Program (Appendix)

7. ____ Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
   a. ___ Computer Readable Copy
   b. ___ Paper Copy (identical to computer copy)
   c. ___ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. __X__ Assignment Papers (cover sheet & documents(s))

9. __X__ a. 37 CFR 3.73(b) Statement (where there is an assignee)

__X__ b. Power of Attorney

10. _____ English Translation Document (if applicable)

11. _____ a. Information Disclosure Statement (IDS)/PTO-1449

_____ b. Copies of IDS Citations

12. _____ Preliminary Amendment

13. __X__ Return Receipt Postcard (MPEP 503) (Should be specifically itemized)

14. _____ a. Small Entity Statement(s)

_____ b. Statement filed in prior application, Status still proper and desired

15. _____ Certified Copy of Priority Document(s) (if foreign priority is claimed)

16. _____ Other: _____

_____

17. **If a CONTINUING APPLICATION,** check appropriate box and supply the requisite information:

___ Continuation ____ Divisional ____ Continuation-in-part (CIP)

of prior application No:_____

18. **Correspondence Address**

_____ Customer Number or Bar Code Label _____
(Insert Customer No. or Attach Bar Code Label here)

or

__X__ Correspondence Address Below

NAME _____

Kurt P. Leyendecker Reg. No. 42,799

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

12400 Wilshire Boulevard 7th Floor, Los Angeles, California 90025

(303) 740-1980 Telephone (303) 740-6962 Facsimile

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: EL580090156US

Date of Deposit May 30, 2000

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner of Patents and Trademarks, Washington, D. C. 20231

Cindy Bennett
(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee)

5/30/00
(Date signed)

UNITED STATES PATENT APPLICATION
FOR

METHOD AND APPARATUS FOR AUTOMATING TESTING OF JAVA BEANS

INVENTORS:

GARY BARNETT
a citizen of USA,
residing at 1491 Cresthaven
San Jose, CA 95135

VAISHALI GHANWAT
a citizen of India,
residing at 1876 Grand Teton Drive
Milpitas, CA 95035

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026
(303) 740-1980

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: EL580090156US
Date of Deposit: May 30, 2000
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner of Patents and Trademarks, Washington, D. C. 20231

Cindy Bennett
(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee)

5/30/00
(Date signed)

METHOD AND APPARATUS FOR AUTOMATING TESTING OF JAVA BEANS

FIELD OF THE INVENTION

This invention relates to software testing in general, and more specifically to a tool for automating the testing of any given Java Bean.

BACKGROUND OF THE INVENTION

Object oriented programming languages such as Java are frequently used in a variety of applications. These applications include programs for use on personal computers as well as complex systems such as those used in Computer Telephony Integration (CTI) networks. In a CTI system, applications written using Java interface with a CTI server to interact with a switch or Automatic Call Distributor (ACD). Object oriented languages such as Java are popular for use in such applications because of their ease of maintenance and expandability. However, testing of new or revised software is a time consuming and therefore expensive process.

In order to test a new or revised program, a programmer is generally required to write a test script or test program which is specific to a particular switch. Such a script or program may include commands to test the functionality of a switch, and possibly provide means for outputting results of the test. The technique of writing a switch specific testing program is time consuming and therefore costly.

SUMMARY OF THE INVENTION

According to one aspect of the invention, a method and system for testing a first program is provided. First, an input file containing a location of a first program, identifiers of other programs to be invoked by the first program, and arguments to be passed to the other programs is read. Next, the first program is executed including invoking the other programs and passing the arguments to those programs. Finally, log files are generated based on results of execution of the first program and the other programs.

BRIEF DESCRIPTION OF THE DRAWINGS

The appended claims set forth the features of the invention with particularity. The invention, together with its advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

Figure 1 is an example of a typical computer system upon which one embodiment of the present invention may be implemented;

Figure 2 is a block diagram illustrating a conceptual view of a Java Bean testing tool for a computer system according to one embodiment of the present invention;

Figure 3 is an example of a typical computer telephony integration network system upon which one embodiment of the present invention may be implemented;

Figure 4 is a block diagram illustrating a conceptual view of a Java Bean testing tool for a computer telephony integration server according to one embodiment of the present invention;

Figure 5 is an example of an input file according to one embodiment of the present invention;

Figure 6 is an example of a bean event log file according to one embodiment of the present invention;

Figure 7 is an example of a script execution log file according to one embodiment of the present invention;

Figure 8 is a flow chart illustrating a Java Bean testing process according to one embodiment of the present invention;

Figure 9 is a flow chart of a class for handling events according to one embodiment of the present invention;

Figure 10 is a flow chart of a class for adding an object according to one embodiment of the present invention;

Figure 11 is a flow chart of a class for loading Java Beans according to one embodiment of the present invention;

Figure 12 is a flow chart of a class for running Java Beans according to one embodiment of the present invention;

Figure 13 is a flow chart of a class for reading an input file according to one embodiment of the present invention;

Figure 14 is a flow chart of a class for writing a log file according to one embodiment of the present invention; and

Figure 15 is a flow chart of a class for creating a user defined object according to one embodiment of the present invention.

DETAILED DESCRIPTION

According to one embodiment of the present invention, methods and systems for testing software are provided. These methods allow testing of a program without requiring an additional test program to be written. Further, no changes to the programming to be tested are required. According to one embodiment of the present invention, an automatic testing tool for generic Java Beans is provided. The tool tests any Java Bean given the path to this Bean and a list of APIs to be invoked.

According to one embodiment of the present invention, an input file containing a location of a first program, identifiers of other programs to be invoked by the first program, and arguments to be passed to the other programs is read. Next, the first program is executed including invoking the other programs and passing to those programs the arguments specified. Finally, log files are generated based on results of execution of the first program and the other programs.

According to one embodiment of the present invention, a Java Bean and a list of APIs and their parameters is read by the testing tool. Next, the Java Bean is located from the given path at runtime. If it does not exist at the given location an error is produced. The tool also checks to see if the given list of APIs exist in the Java Bean. If so, the given APIs are executed in the order they appear in the input file. The tool also recognizes some other keywords not existing in the Java Bean such as "sleep" and "loop".

According to another aspect of the present invention, special commands can be included in the input file. Generally, these commands can be used to control the flow of program execution. For example, execution of the program to be tested can be caused to

repeat some code or to temporarily pause execution. Details of these commands and other aspects of the present invention will be described further below.

In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and devices are shown in block diagram form.

The present invention includes various operations, which will be described below. The operations of the present invention may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor or logic circuits programmed with the instructions to perform the operations. Alternatively, the operations may be performed by a combination of hardware and software.

The present invention may be provided as a computer program product which may include a machine-readable medium having stored thereon instructions which may be used to program a computer (or other electronic devices) to perform a process according to the present invention. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnet or optical cards, flash memory, or other type of media / machine-readable medium suitable for storing electronic instructions. Moreover, the present invention may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer to a requesting

computer by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

Importantly, while embodiments of the present invention will be described with reference to automated testing of Java Beans interacting with a Computer-Telephony Integration (CTI) server, the method and apparatus described herein are equally applicable to automated testing of other types of software on different types of platforms. For example, the techniques described herein are thought to be useful in connection with automating testing of other object oriented languages in systems such as personal computers. For example, similar techniques can be applied to automated testing of a DLL or an ActiveX control.

Figure 1 is an example of a typical computer system upon which one embodiment of the present invention may be implemented. Computer system 100 comprises a bus or other communication means 101 for communicating information, and a processing means such as processor 102 coupled with bus 101 for processing information. Computer system 100 further comprises a random access memory (RAM) or other dynamic storage device 104 (referred to as main memory), coupled to bus 101 for storing information and instructions to be executed by processor 102. Main memory 104 also may be used for storing temporary variables or other intermediate information during execution of instructions by processor 102. Computer system 100 also comprises a read only memory (ROM) and/or other static storage device 106 coupled to bus 101 for storing static information and instructions for processor 102.

A data storage device 107 such as a magnetic disk or optical disc and its

corresponding drive may also be coupled to computer system 100 for storing information and instructions. Computer system 100 can also be coupled via bus 101 to a display device 121, such as a cathode ray tube (CRT) or Liquid Crystal Display (LCD), for displaying information to an end user. Typically, an alphanumeric input device 122, including alphanumeric and other keys, may be coupled to bus 101 for communicating information and/or command selections to processor 102. Another type of user input device is cursor control 123, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 102 and for controlling cursor movement on display 121.

A communication device 125 is also coupled to bus 101. The communication device 125 may include a modem, a network interface card, or other well known interface devices, such as those used for coupling to Ethernet, token ring, or other types of physical attachment for purposes of providing a communication link to support a local or wide area network, for example. In this manner, the computer system 100 may be coupled to a number of clients and/or servers via a conventional network infrastructure, such as a company's Intranet and/or the Internet, for example.

It is appreciated that a lesser or more equipped computer system than the example described above may be desirable for certain implementations. Therefore, the configuration of computer system 100 will vary from implementation to implementation depending upon numerous factors, such as price constraints, performance requirements, technological improvements, and/or other circumstances.

It should be noted that, while the operations described herein may be performed

under the control of a programmed processor, such as processor 102, in alternative embodiments, the operations may be fully or partially implemented by any programmable or hardcoded logic, such as Field Programmable Gate Arrays (FPGAs), TTL logic, or Application Specific Integrated Circuits (ASICs), for example. Additionally, the method of the present invention may be performed by any combination of programmed general purpose computer components and/or custom hardware components. Therefore, nothing disclosed herein should be construed as limiting the present invention to a particular embodiment wherein the recited operations are performed by a specific combination of hardware components.

Figure 2 is a block diagram illustrating a conceptual view of a Java Bean testing tool for a computer system according to one embodiment of the present invention. Here, the Java Bean testing tool 202 reads an input file 201. This input file 201 includes the pathname to the program or Java Bean to be tested, names of other programs to be invoked by the program being tested, and the arguments to be passed to the programs to be invoked. The programs to be invoked may be part of an Application Program Interface (API). Additionally, this file 201 may contain special instructions to the testing tool. These instructions as well as the other contents of this file will be discussed in greater detail below with reference to Figure 5.

After reading the input file, the Java Bean testing tool 202 executes the program to be tested including invoking programs listed in the input file, passing to those programs the arguments specified, and performing any special instruction such as "loop" and "sleep" included in the input file.

Finally, the Java Bean testing tool 202 generates a results log 203. This results log 203 contains information based on results of execution of the program to be tested and the programs to be invoked by the program to be tested.

Alternatively, the present invention may be implemented on a computer telephony integration (CTI) system. Figure 3 is an example of a typical CTI system upon which one embodiment of the present invention may be implemented. In this system the Java Bean tool loads a Java Bean which communicates with a CTI server 301. Typically, this server will include many of the same features as the typical computer system described above with reference to Figure 1. For example, the CTI server 301 will likely include a processor 102, bus 101, main memory 104, read only memory 106, mass storage devices 107, a display 121, a keyboard 122, a cursor control device 123, and a communication device 125. In the CTI server 301, the communication device 125 is likely to be a network adapter. An example of equipment that can be used as a CTI server is the Aspect Communications Office Telephony Server produced by Aspect Communications, Inc.

The CTI server 301 is connected to an Automatic Call Distributor (ACD)/switch/PBX 302. The ACD switch 302 controls and routes all the calls using call control tables. Equipment that can be used as an ACD switch includes various standard ACD switches manufactured by Aspect Communications, Inc. and others. This switch 302 is then connected to various CTI network devices such as telephones 303, fax machines 304, or email 305. An application written using a Java Bean can send telephony commands to a CTI server which interprets these commands into a proprietary language understood by the switch.

As explained above, the present invention may be implemented in a variety of systems. In some cases the present invention may be implemented in a CTI system such as that described with reference to Figure 3. Figure 4 is a block diagram illustrating a conceptual view of a Java Bean testing tool in a CTI system according to one embodiment of the present invention. Here, the Java Bean testing tool 402 reads an input file 401. This input file 401 includes the pathname to the program or Java Bean to be tested, names of other programs to be invoked by the program being tested, and arguments to be passed to the programs to be invoked. Further, this file 401 may contain special commands for the testing tool 402. These commands along with the other contents of this file will be discussed in greater detail below with reference to Figure 5.

After reading the input file, the Java Bean testing tool 402 executes the program to be tested including invoking programs listed in the input file, passing to those programs the arguments specified, and executing any special commands contained in the input file. The programs to be invoked may be part of an Application Program Interface (API). For example, in a Microsoft Windows environment the programs to be invoked may consist of .dll type files.

Finally, the Java Bean testing tool 402 generates result logs 403 and 404. These logs can include a script execution log 403 and a bean event log 404. According to one embodiment of the present invention, when implemented on a CTI server, the script execution log 403 can contain responses from an ACD switch. In such a system the contents of the script execution log are based on the results of execution of the program to be tested and responses from an ACD switch. Similarly, when implemented on a CTI

server, the bean event log 404 may contain results from one or more APIs. In such a system the bean event log 404 contains information based on results of execution of the program to be tested and the programs to be invoked by the program to be tested along with date and time stamps for each entry.

Figure 5 is an example of an input file according to one embodiment of the present invention. In this file, comment lines 500 may be identified with a pound sign (#) as the first character of the line. This file contains the name of the program to be tested 505 and a list of programs to be invoked during execution of the program being tested and parameters to be passed to these programs 510.

Further, this file may contain special commands to the testing tool such as a loop command 515 or a sleep command 520. The loop command 515 instructs the tool to repeat all instructions between the loop command 515 and the subsequent endloop command 525. In this example, the sleep command and the MakeCall command located between the loop command 515 and the endloop command 525 are repeated 4 times. The sleep command 520 instructs the tool to pause for the specified time period in seconds. Such a command may be useful for pausing the test in order to wait for a response after invoking a program. In this example, a sleep command 520 is used to pause execution after a MakeCall command.

Figure 6 is an example of a bean event log file according to one embodiment of the present invention. The contents of such a log are based on the results of execution of an API to be tested and any programs to be invoked by the program being tested. In this example, each entry in the log is time stamped 605 and a result 610 is recorded. These

results 610 include the name of the API as well as the value returned. For example, the first entry 615 is the result of a Method1 620 instruction which returned a value of null 625.

Figure 7 is an example of a script execution log file according to one embodiment of the present invention. When implemented in a CTI system, the contents of such a log are based on the responses from the switch corresponding to the APIs invoked. In this example, each entry in the log is time stamped 705 and a result 710 is recorded. These results 710 can indicate various responses by the ACD switch to commands issued to the switch. In this example, properties of events such as message type 715, status indicator 720, and status 725 as well as others are indicated.

Figure 8 is a flow chart illustrating a Java Bean testing process according to one embodiment of the present invention. First 810, the input file is read line by line. If the specified Java Bean does not exist 815, no further processing is performed. If the specified Java Bean is located 815, the bean is loaded 820. Once the specified bean has been loaded, the API names and parameters are read from the input file 825. Each API and parameter read are then stored in an object 830 and each of these objects are then accessed 835. If the APIs do not exist in the bean 840, that is, an instance of the API object has not been created, no further processing occurs. If the APIs do exist in the bean 840, they are accessed and invoked in the order in which they appear 845. If the input file does not specify an event listener 850, a script execution log is created but is left empty. If an event listener is specified 850, events are logged in the script execution log 855 and the bean event log 860 and each entry is date and time stamped 865.

According to one embodiment of the present invention, the methods described above can be implemented using an object-oriented language such as Java. In one such implementation, several classes can be created and used. These classes may include a class for handling events (an ActionEvent class), a class for adding an object (an AddObject class), a class for loading Java Beans (a JarClassLoader class), a class for running Java Beans (a JarRunner class), a class for reading an input file (a LoadJarFile class), a class for writing a log file (a Logfile class), and a class for creating a user defined object (a UserDefinedObject class). Each of these classes will be described below with reference to Figures 9 through 15.

Figure 9 is a flow chart of a class for handling events (an ActionEvent class) according to one embodiment of the present invention. This class first loads the specified class and passes it to an action listener method 905. Next, all event handling routines are written into the class 910. Finally, when an event occurs, the events and properties are logged with a date and time stamp 915.

Figure 10 is a flow chart of a class for adding an object (an AddObject class) according to one embodiment of the present invention. If the object to be added is an Application Program Interface (API) 1005, this class stores the name and parameters of the API 1010. If the object to be added is a user defined class 1015, this class stores the user-defined class in a separate list 1020.

Figure 11 is a flow chart of a class for loading Java Beans (a JarClassLoader class) according to one embodiment of the present invention. This class first gets the bean class name from the URI 1105. Next, this class creates an instance of the bean from the

class name 1110. The listener class is then retrieved, an instance of the class is created and the instance is passed to an action listener 1115. The class then extracts API information from the add object object, invokes the API and stores the results 1120. Finally, the API results are logged with a date and time stamp 1125.

Figure 12 is a flow chart of a class for running Java Beans (a JarRunner class) according to one embodiment of the present invention. This class first loads the bean class 1205. Next, the listener class is loaded 1210. Finally, the class invokes all APIs listed in the input file 1215.

Figure 13 is a flow chart of a class for reading an input file (a LaodJarFile class) according to one embodiment of the present invention. This class first reads a file name from the command file 1305. This file is then parsed to read the bean name, listener class name, APIs and API parameters 1310. The word read is identified as either a "BeanName" 1315, "Loadlistener" 1325, "MethodName" 1335, "Parameters" 1345, "loop" 1355, "endloop" 1365, or "sleep" 1375. If the word read does not match any of these, no processing is performed. If the word read is "BeanName" 1315, the class gets the bean name and uses the name to load the bean class 1320. When the word read is "Loadlistener" 1325, the class gets the listener class name and uses it to load the listener class 1330. If the word read is a "MethodName" 1335, the class creates a new object which stores API information and the API name 1340. When the word read is "Parameters" 1345, the class gets all the parameters of the API and stores them 1350 in an object created in step 1340.

If the word read is "loop" 1355, the class invokes all APIs before loop, creates a list of APIs to be repeated, and stores the loop count 1360. When an "endloop" is read 1365, the class repeats the APIs to be repeated between loop and endloop for the given number of times 1370. If the word read is "sleep" 1375, the class gets the sleep time 1380.

Figure 14 is a flow chart of a class for writing a log file (a Logfile class) according to one embodiment of the present invention. This class logs the given string in a given file name with a date and time stamp 1405.

Figure 15 is a flow chart illustrating a class for creating a user defined object (a UserDefinedObject class) according to one embodiment of the present invention. This class stores information about a user defined object such as class name and member name 1505.

CLAIMS

What is claimed is:

1. A method of testing a first program comprising:

reading an input file containing a location of a first program, identifiers of other programs to be invoked by the first program, and arguments to be passed to the other programs;

executing the first program including invoking the other programs and passing the arguments to the other programs; and

generating one or more log files based on results of execution of the first program.

2. The method of claim 1, where said input file further includes special commands.

3. The method of claim 2, where said special commands include:

a loop command that causes execution of the first program to repeat, for a specified number of iterations, subsequent instructions until the occurrence of an endloop instruction; and

a sleep command that causes execution of the first program to pause for a specified time.

4. The method of claim 1, where said input file is a text file.

5. The method of claim 1, where said first program is a Java Bean.

6. The method of claim 1, where said other programs are an Application Program Interface (API).

7. The method of claim 1, where executing the first program further comprises sending commands to an Automatic Call Distributor (ACD) switch through a CTI server.

8. The method of claim 1, where said log file is a text file.

9. The method of claim 1, where each entry in said log file is date and time stamped.

10. The method of claim 1, where contents of one of said one or more log files are based on results of execution of the first program and the other programs.

11. The method of claim 1, where contents of one of said one or more log files are based on results of execution of the first program and responses from an Automatic Call Distributor (ACD) switch.

12. The method of claim 1, further comprising checking results of program testing by:

generating one or more files containing expected results of execution of the first program and the other programs; and

comparing one or more of the files containing expected results to one or more of said one or more log files.

13. A system comprising:

a storage device having stored therein a routine for testing a first program; and

a processor coupled to the storage device to execute the routine for testing of the first program such that execution of said routine causes the processor to:

read an input file containing a location of the first program, identifiers of other programs to be invoked by the first program, and arguments to be passed to the other programs,

execute the first program including invoking the other programs and passing the arguments to the other programs, and

generate one or more log files based on results of execution of the first program.

14. The system of claim 13, where said input file further includes special commands.

15. The system of claim 14, where said special commands include:

a loop command that causes execution of the first program to repeat, for a specified number of iterations, subsequent instructions until the occurrence of an endloop instruction; and

a sleep command which causes execution of the first program to pause for a specified time.

16. The system of claim 13, where said input file is a text file.

17. The system of claim 13, where said first program is a Java Bean.

18. The system of claim 13, where said other programs are part of an Application Program Interface (API).

19. The system of claim 13, where execution of the first program further comprises making calls to an Automatic Call Distribution (ACD) switch.

20. The system of claim 13, where said log file is a text file.

21. The system of claim 13, where each entry in said log file is date and time stamped.

22. The system of claim 13, where contents of said one or more log files are based on results of execution of the first program and the other programs.

23. The system of claim 13, where contents of said one or more log files are based on results of execution of the first program and responses from an Automatic Call Distributor (ACD) switch.

24. The system of claim 13, further comprising instructions for checking results of program testing such that execution of the instructions causes the processor to:

generate one or more files containing expected results of execution of the first program and the other programs; and

compare one or more of the files containing expected results to one or more of the log files.

25. A computer readable medium having stored thereon instructions which, when executed by a general purpose machine, test a first program by causing the general purpose machine to:

read an input file containing a location of the first program, identifiers of other programs to be invoked by the first program, and arguments to be passed to the other programs;

execute the first program including invoking the other programs listed and passing the arguments to the other programs; and

generate one or more log files based on results of execution of the first program.

26. The computer readable medium of claim 25, where said input file further includes special commands.

27. The computer readable medium of claim 26, where said special commands include:

a loop command which causes execution of the first program to repeat for a specified number of iterations all subsequent instructions until the occurrence of an endloop instruction; and

a sleep command which causes execution of the first program to pause for a specified time.

28. The computer readable medium of claim 25, where said input file is a text file.

29. The computer readable medium of claim 25, where said first program is a Java Bean.

30. The computer readable medium of claim 25, where said other programs are part of an Application Program Interface (API).

31. The computer readable medium of claim 25, where executing the first program further comprises making calls to an Automatic Call Distributor (ACD) switch.

32. The computer readable medium of claim 25, where said log file is a text file.

33. The computer readable medium of claim 25, where each entry in said log file is date and time stamped.

34. The computer readable medium of claim 25, where contents of one of said one or more log files are based on results of execution of the first program and the other programs.

35. The computer readable medium of claim 25, where contents of one of said one or more log files are based on results of execution of the first program and responses from an Automatic Call Distributor (ACD) switch.

36. The computer readable medium of claim 25, further comprising instructions for checking results of program testing such that execution of the instructions causes the processor to:

generate one or more files containing expected results of execution of the first program and the other programs; and

compare one or more of the file containing expected results to one or more of said one or more log files.

ABSTRACT

According to the invention, systems, and methods are disclosed for testing of a first program is provided. First, an input file containing a location of a first program, identifiers of other programs to be invoked by the first program, and arguments to be passed to the other programs is read. Next, the first program is executed including invoking the other programs and passing to those programs the arguments specified. Finally, log files are generated based on results of execution of the first program and the other programs.

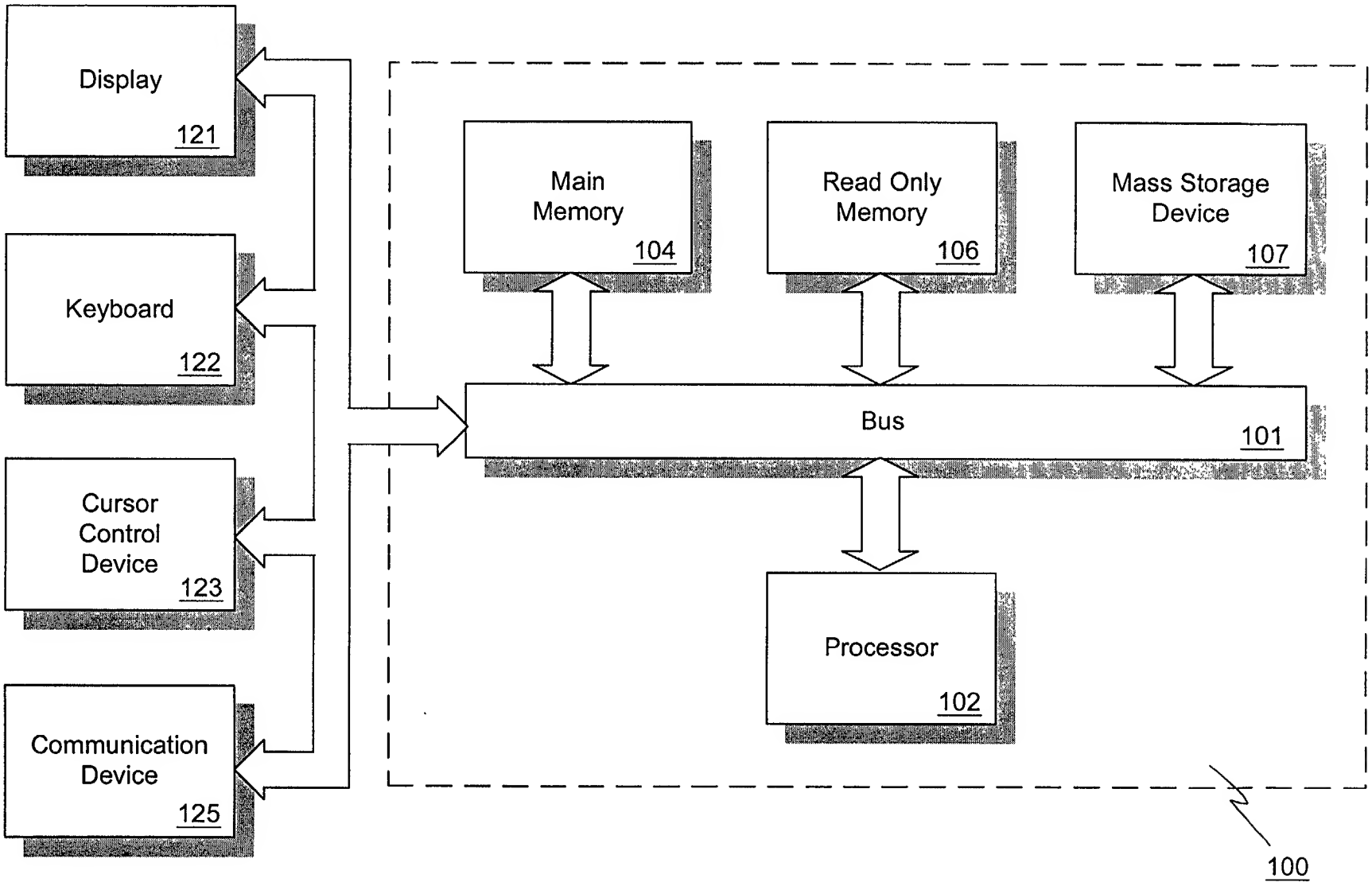
Display
121
Keyboard
122
Cursor
Control
Device
123
Communication
Device
125
Main
Memory
104
Read Only
Memory
106
Mass Storage
Device
107
Bus
101
Processor
102
100

Figure 1

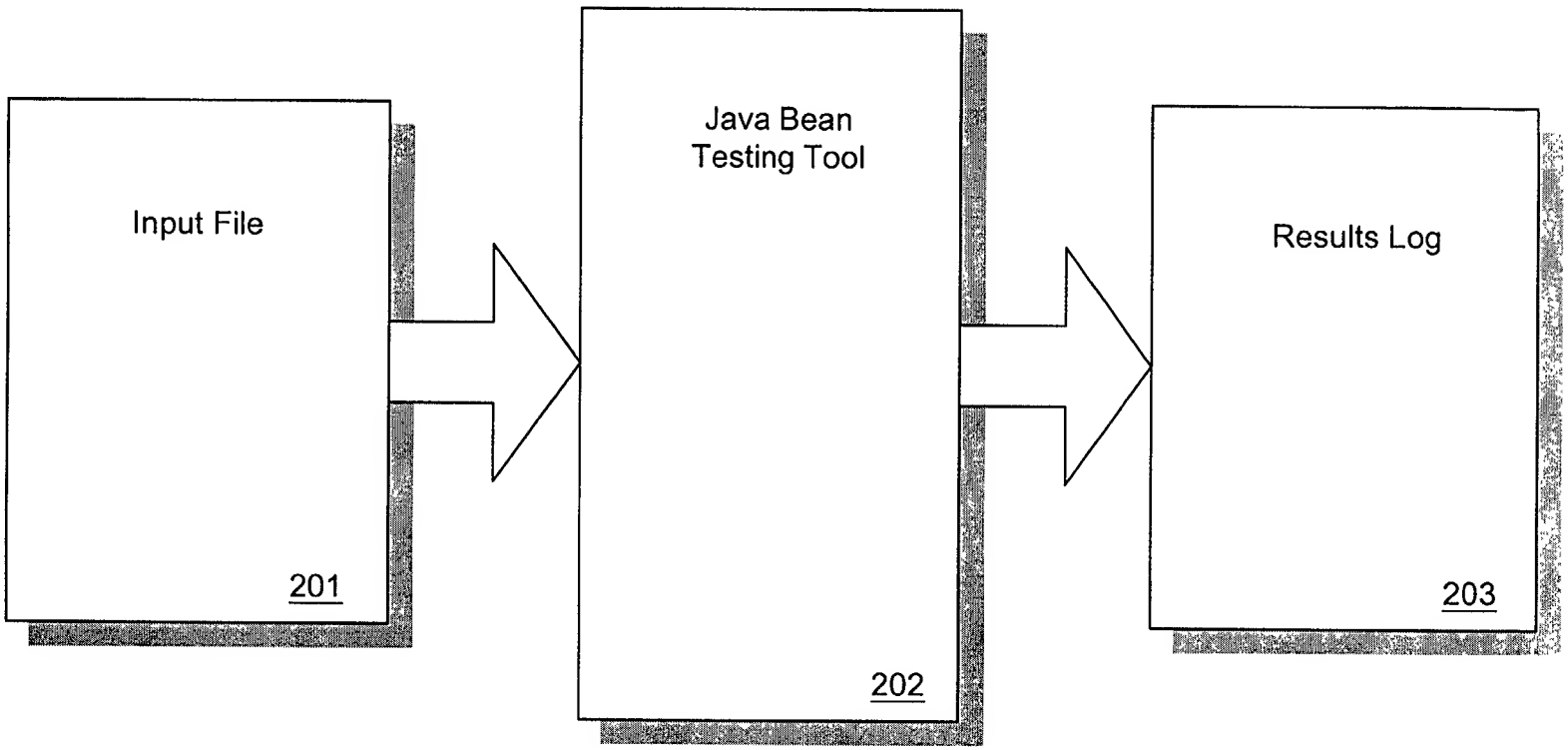

Figure 2

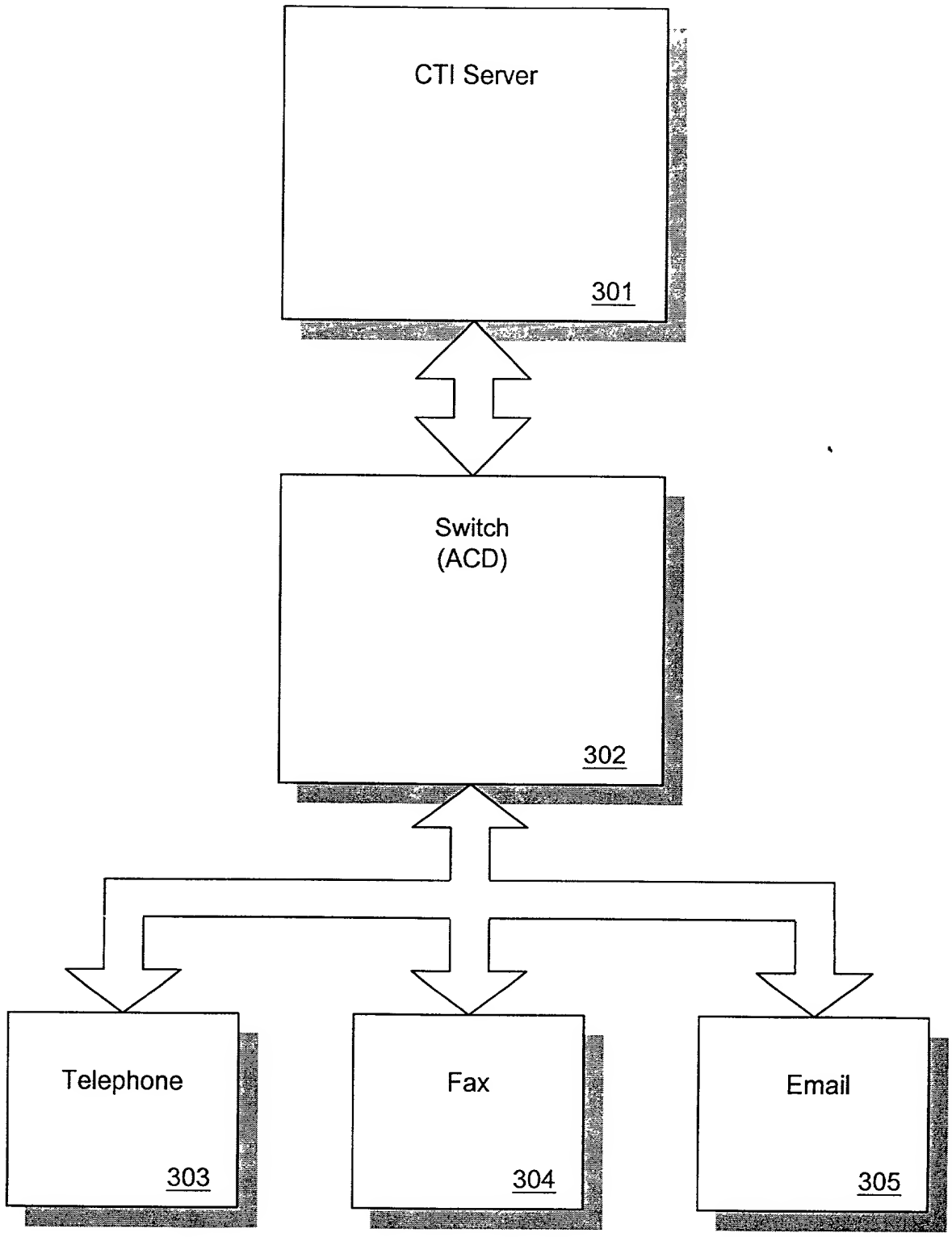
CTI Server
301
Switch
(ACD)
302
Telephone
303
Fax
304
Email
305

Figure 3

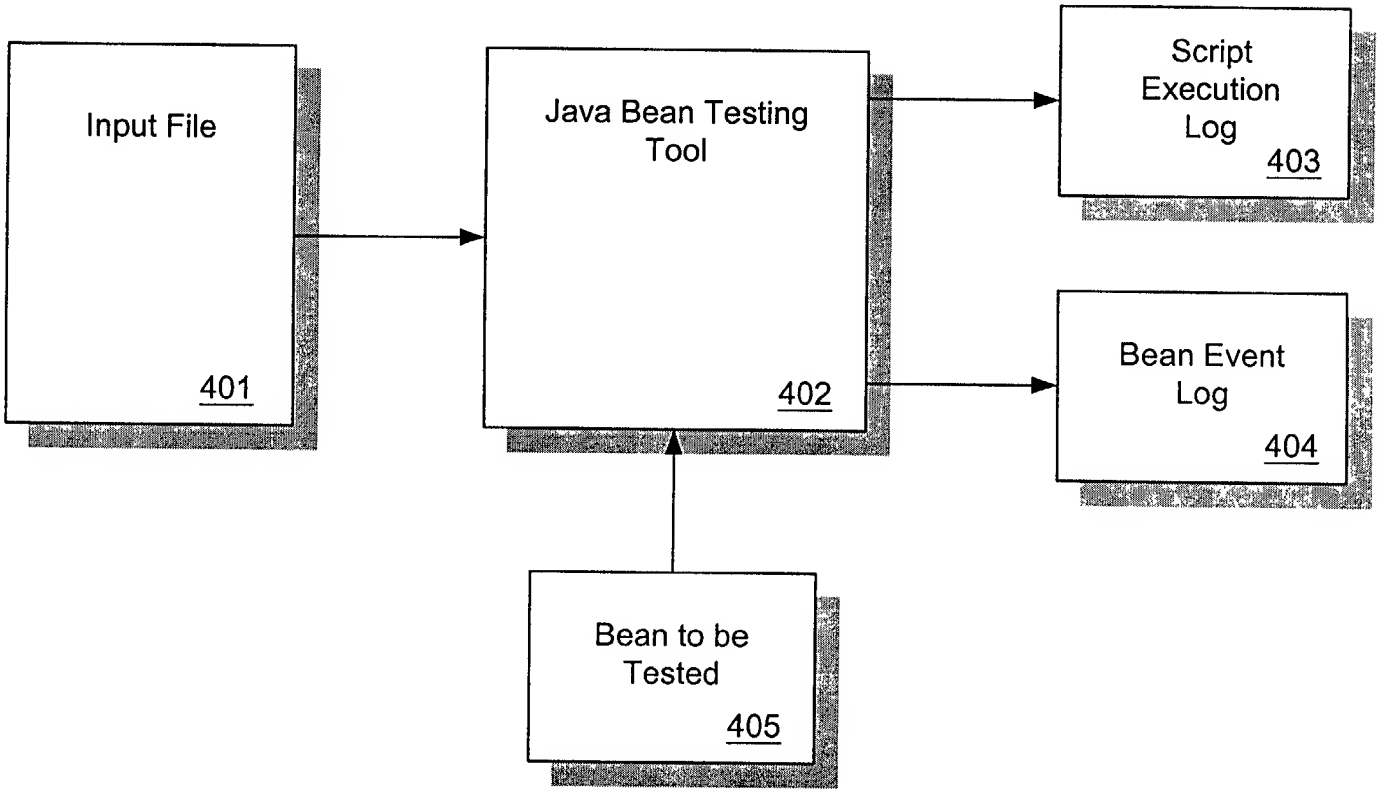
Input File
401
Java Bean Testing Tool
402
Script Execution Log
403
Bean Event Log
404
Bean to be Tested
405

Figure 4

```
# Descriptor should be one word
# This file has path to a bean and it also
# has all the APIs to be implemented

BeanName = D:/Aspect/nortel/java/MLObject.jar
MethodName = Method1
Parameters = vghanwat-pc
MethodName = Method2
Parameters = int 9001
MethodName = Method3
Parameters = NONE
MethodName = Method4
Parameters = NONE
MethodName = Method5
Parameters = NONE
MethodName = Method6
Parameters = short 8 string 4040 string 5146
MethodName = Method7
Parameters = short 8 string 4040
MethodName = Method8
Parameters = short 1

loop 4
sleep 5
MethodName = Method9
Parameters = short 8 string 5159 short 8 string 7007 short 0
endloop

sleep 5
MethodName = Method10
Parameters = short 8 string 4040
# MethodName = Method11
# Parameters = NONE
```

500, 505, 510, 515, 520, 525

Figure 5

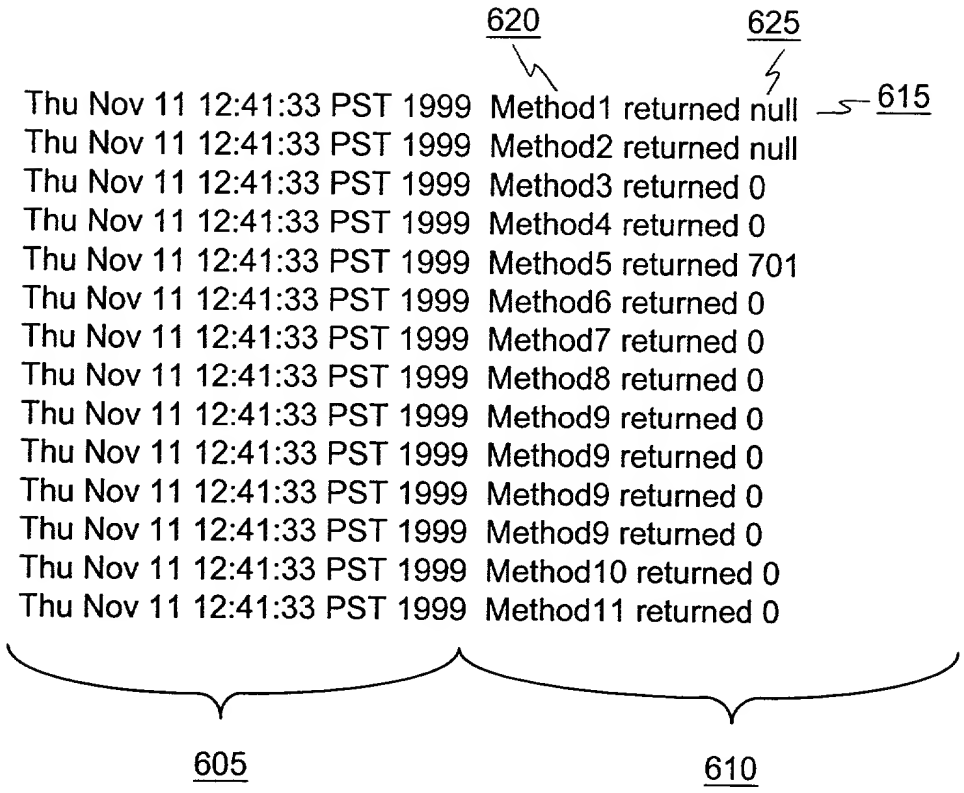
620
625
Thu Nov 11 12:41:33 PST 1999 Method1 returned null
615
Thu Nov 11 12:41:33 PST 1999 Method2 returned null
Thu Nov 11 12:41:33 PST 1999 Method3 returned 0
Thu Nov 11 12:41:33 PST 1999 Method4 returned 0
Thu Nov 11 12:41:33 PST 1999 Method5 returned 701
Thu Nov 11 12:41:33 PST 1999 Method6 returned 0
Thu Nov 11 12:41:33 PST 1999 Method7 returned 0
Thu Nov 11 12:41:33 PST 1999 Method8 returned 0
Thu Nov 11 12:41:33 PST 1999 Method9 returned 0
Thu Nov 11 12:41:33 PST 1999 Method9 returned 0
Thu Nov 11 12:41:33 PST 1999 Method9 returned 0
Thu Nov 11 12:41:33 PST 1999 Method9 returned 0
Thu Nov 11 12:41:33 PST 1999 Method10 returned 0
Thu Nov 11 12:41:33 PST 1999 Method11 returned 0
605
610

Figure 6

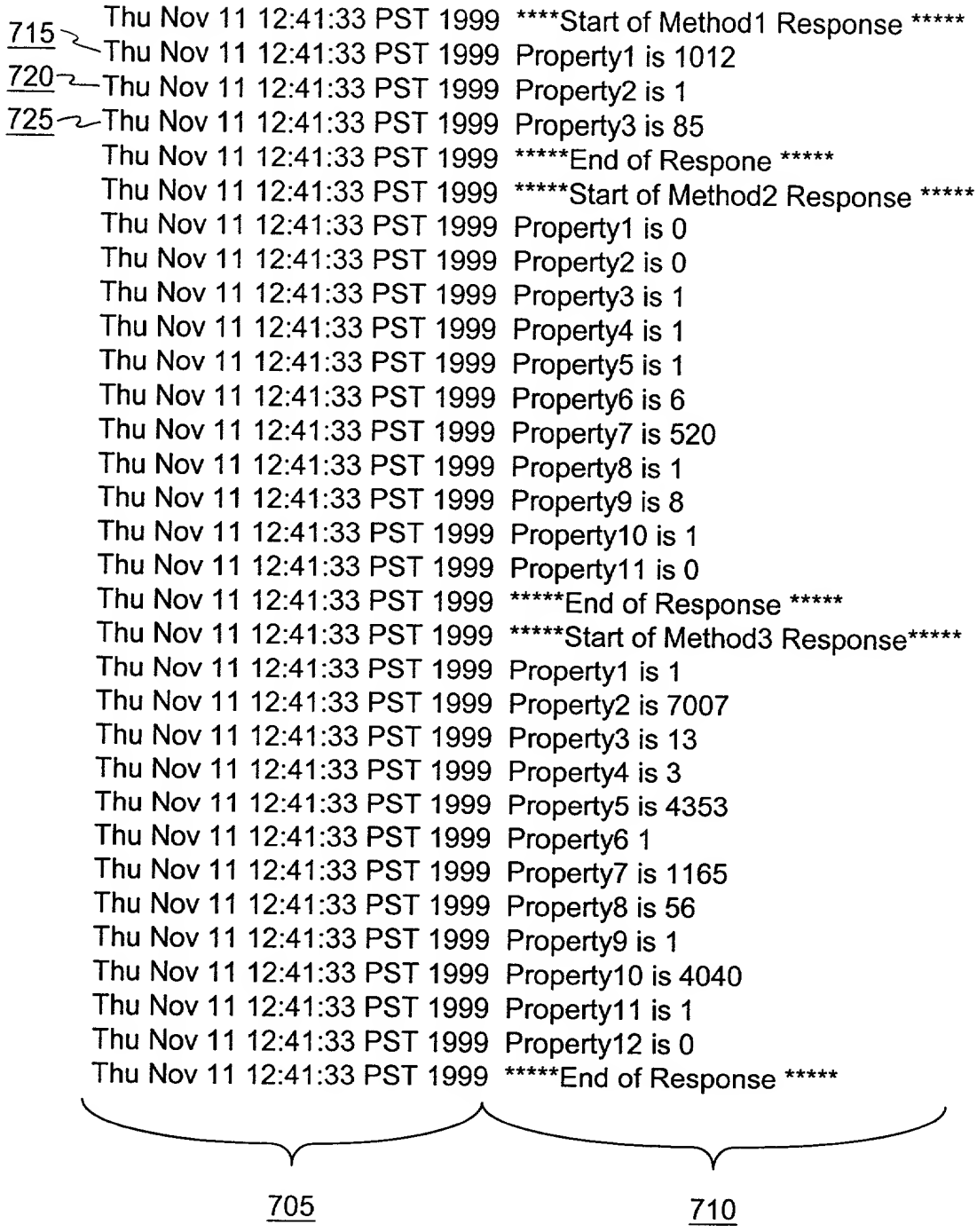
Thu Nov 11 12:41:33 PST 1999 ****Start of Method1 Response *****
715
Thu Nov 11 12:41:33 PST 1999 Property1 is 1012
720
Thu Nov 11 12:41:33 PST 1999 Property2 is 1
725
Thu Nov 11 12:41:33 PST 1999 Property3 is 85
Thu Nov 11 12:41:33 PST 1999 *****End of Respone *****
Thu Nov 11 12:41:33 PST 1999 *****Start of Method2 Response *****
Thu Nov 11 12:41:33 PST 1999 Property1 is 0
Thu Nov 11 12:41:33 PST 1999 Property2 is 0
Thu Nov 11 12:41:33 PST 1999 Property3 is 1
Thu Nov 11 12:41:33 PST 1999 Property4 is 1
Thu Nov 11 12:41:33 PST 1999 Property5 is 1
Thu Nov 11 12:41:33 PST 1999 Property6 is 6
Thu Nov 11 12:41:33 PST 1999 Property7 is 520
Thu Nov 11 12:41:33 PST 1999 Property8 is 1
Thu Nov 11 12:41:33 PST 1999 Property9 is 8
Thu Nov 11 12:41:33 PST 1999 Property10 is 1
Thu Nov 11 12:41:33 PST 1999 Property11 is 0
Thu Nov 11 12:41:33 PST 1999 *****End of Response *****
Thu Nov 11 12:41:33 PST 1999 *****Start of Method3 Response*****
Thu Nov 11 12:41:33 PST 1999 Property1 is 1
Thu Nov 11 12:41:33 PST 1999 Property2 is 7007
Thu Nov 11 12:41:33 PST 1999 Property3 is 13
Thu Nov 11 12:41:33 PST 1999 Property4 is 3
Thu Nov 11 12:41:33 PST 1999 Property5 is 4353
Thu Nov 11 12:41:33 PST 1999 Property6 1
Thu Nov 11 12:41:33 PST 1999 Property7 is 1165
Thu Nov 11 12:41:33 PST 1999 Property8 is 56
Thu Nov 11 12:41:33 PST 1999 Property9 is 1
Thu Nov 11 12:41:33 PST 1999 Property10 is 4040
Thu Nov 11 12:41:33 PST 1999 Property11 is 1
Thu Nov 11 12:41:33 PST 1999 Property12 is 0
Thu Nov 11 12:41:33 PST 1999 *****End of Response *****
705
710

Figure 7

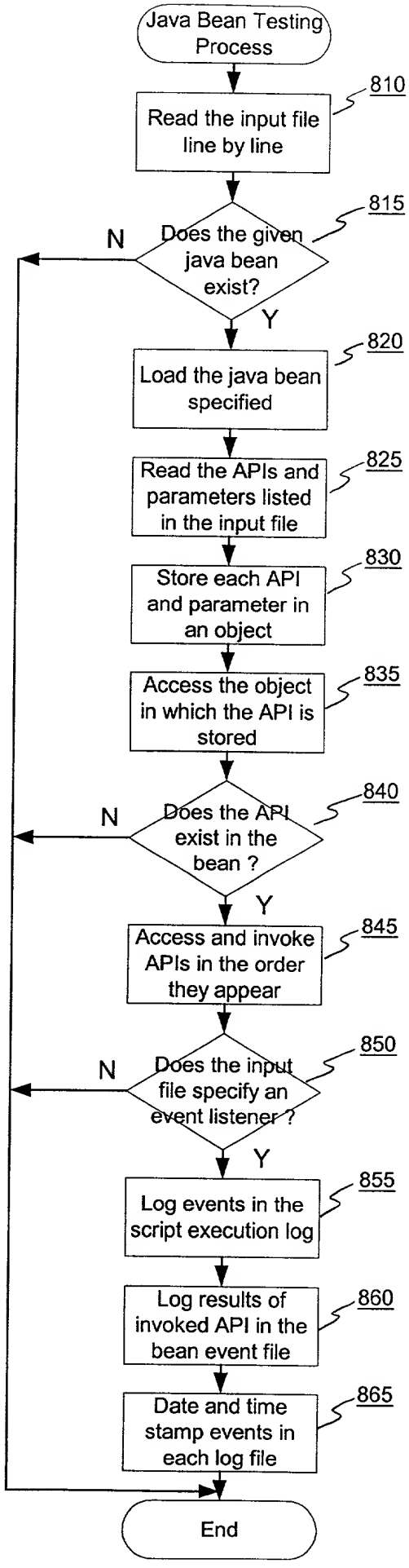
Java Bean Testing Process
Read the input file line by line
810
Does the given java bean exist?
815
N
Y
Load the java bean specified
820
Read the APIs and parameters listed in the input file
825
Store each API and parameter in an object
830
Access the object in which the API is stored
835
Does the API exist in the bean ?
840
N
Y
Access and invoke APIs in the order they appear
845
Does the input file specify an event listener ?
850
N
Y
Log events in the script execution log
855
Log results of invoked API in the bean event file
860
Date and time stamp events in each log file
865
End

Figure 8

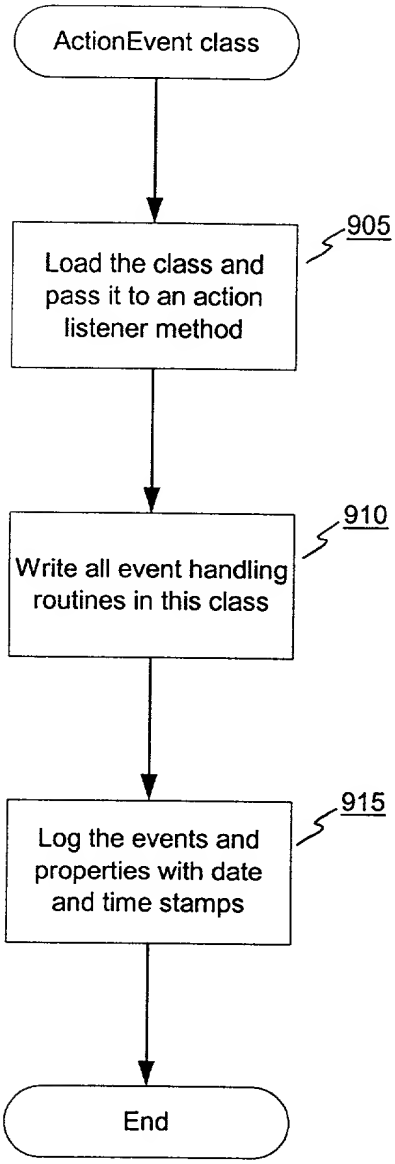
ActionEvent class
Load the class and pass it to an action listener method
905
Write all event handling routines in this class
910
Log the events and properties with date and time stamps
915
End

Figure 9

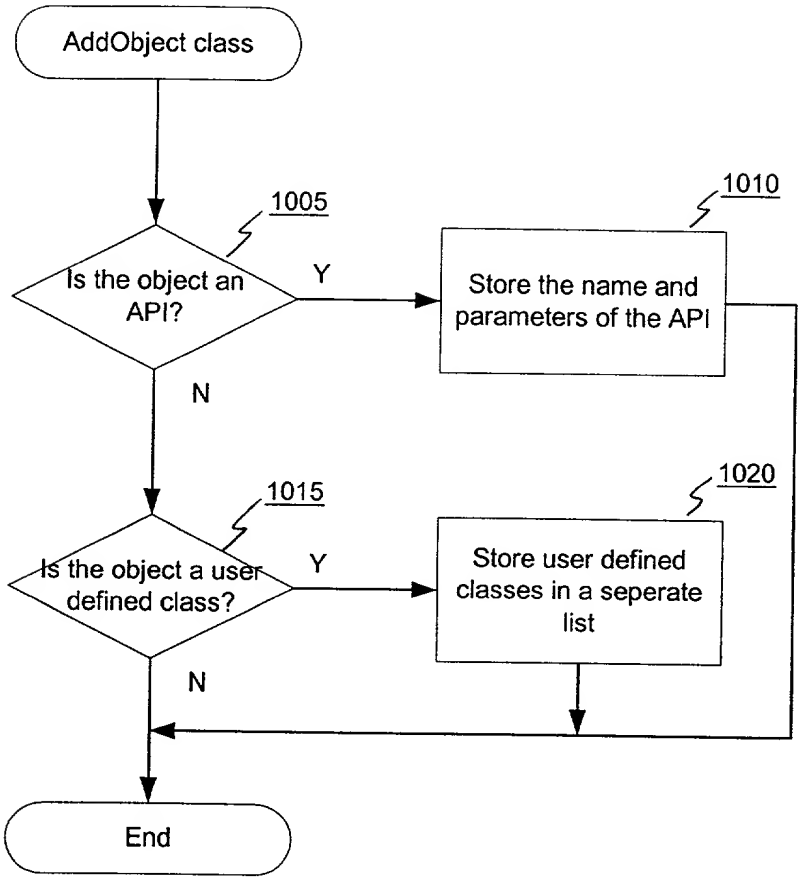
AddObject class
1005
Is the object an API?
Y
1010
Store the name and parameters of the API
N
1015
Is the object a user defined class?
Y
1020
Store user defined classes in a seperate list
N
End

Figure 10

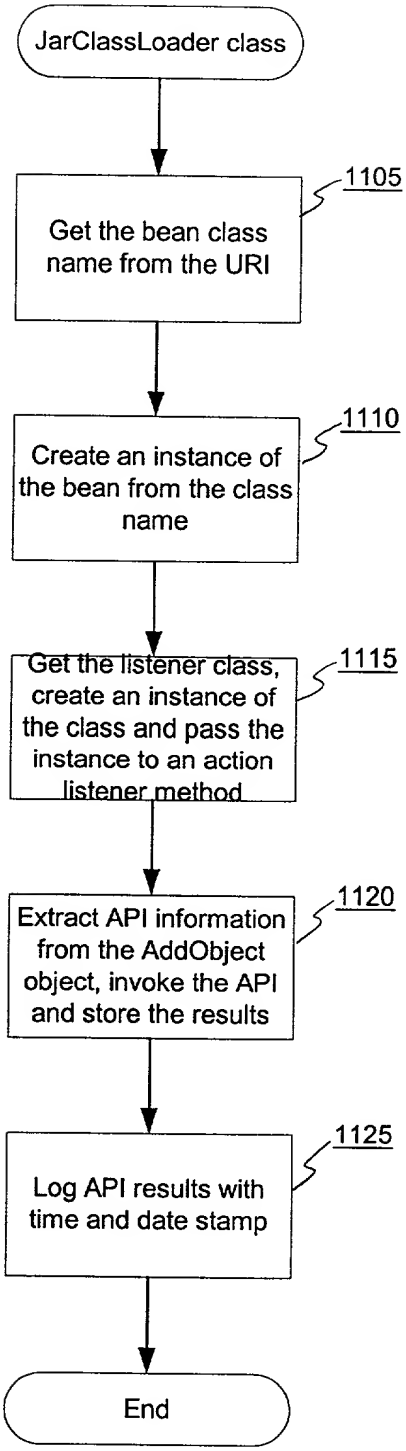
JarClassLoader class
1105
Get the bean class name from the URI
1110
Create an instance of the bean from the class name
1115
Get the listener class, create an instance of the class and pass the instance to an action listener method
1120
Extract API information from the AddObject object, invoke the API and store the results
1125
Log API results with time and date stamp
End

Figure 11

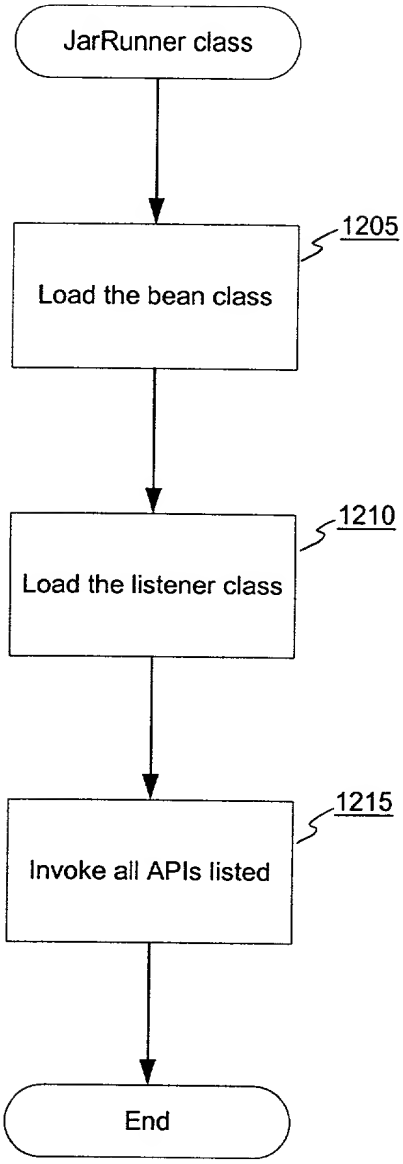
JarRunner class
1205
Load the bean class
1210
Load the listener class
1215
Invoke all APIs listed
End

Figure 12

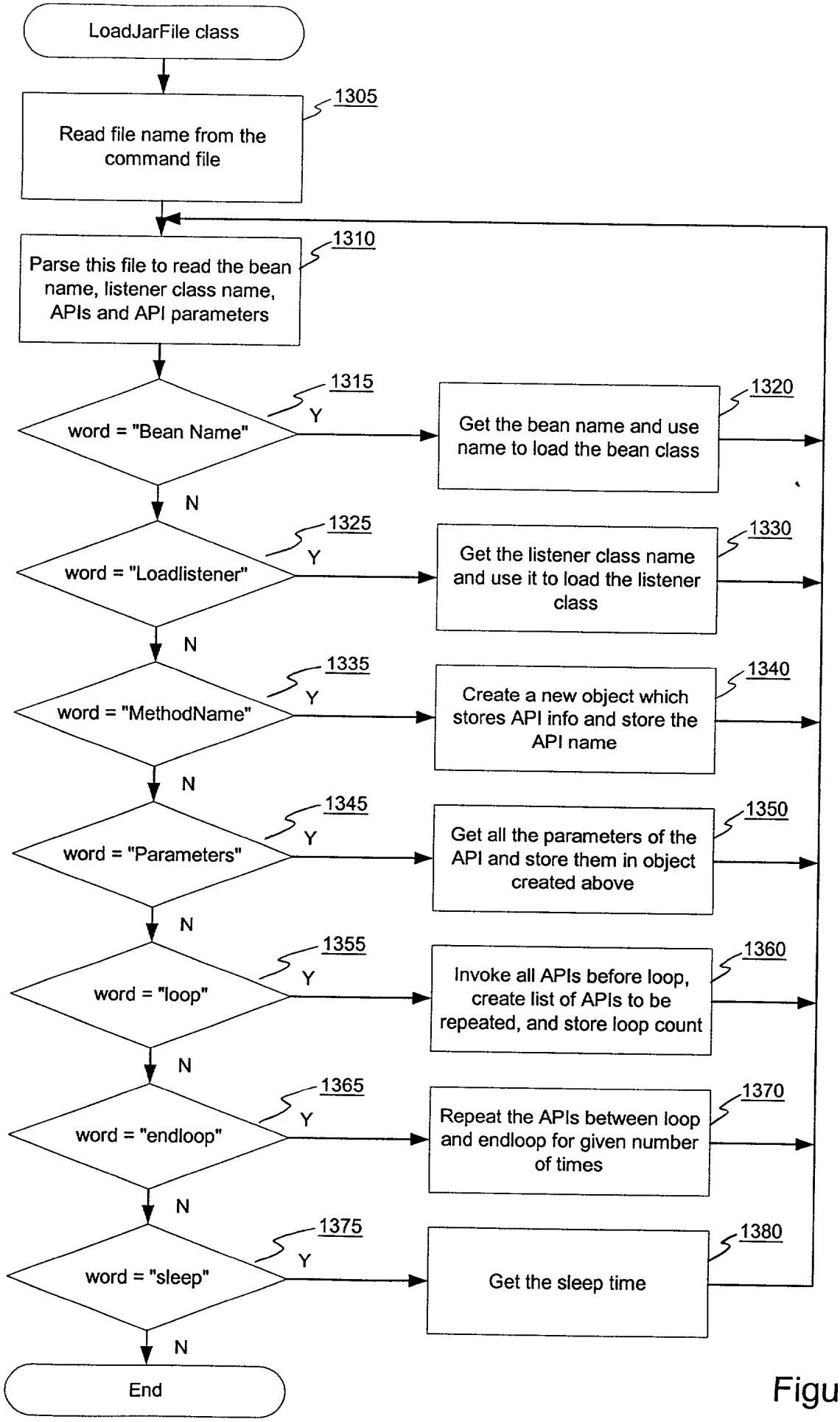
LoadJarFile class
Read file name from the command file
1305
Parse this file to read the bean name, listener class name, APIs and API parameters
1310
word = "Bean Name"
1315
Y
Get the bean name and use name to load the bean class
1320
N
word = "Loadlistener"
1325
Y
Get the listener class name and use it to load the listener class
1330
N
word = "MethodName"
1335
Y
Create a new object which stores API info and store the API name
1340
N
word = "Parameters"
1345
Y
Get all the parameters of the API and store them in object created above
1350
N
word = "loop"
1355
Y
Invoke all APIs before loop, create list of APIs to be repeated, and store loop count
1360
N
word = "endloop"
1365
Y
Repeat the APIs between loop and endloop for given number of times
1370
N
word = "sleep"
1375
Y
Get the sleep time
1380
N
End

Figure 13

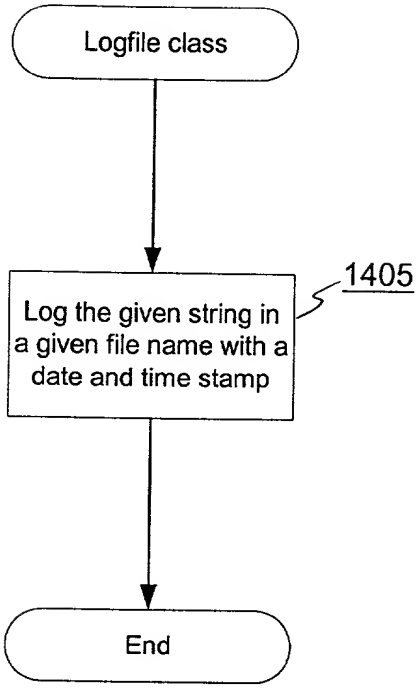
Logfile class
1405
Log the given string in a given file name with a date and time stamp
End

Figure 14

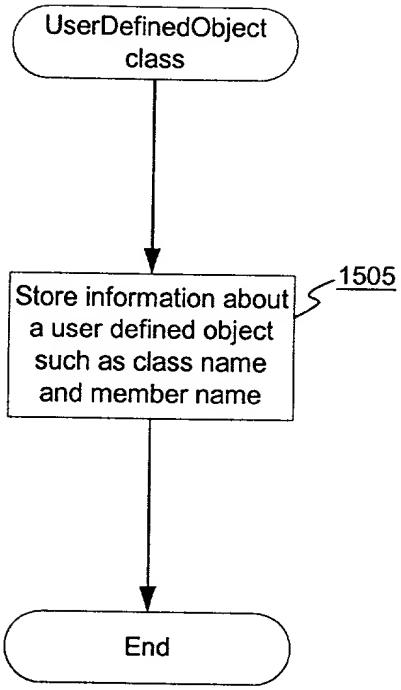
UserDefinedObject class
Store information about a user defined object such as class name and member name
1505
End

Figure 15

Attorney's Docket No.: 002950.P055 Patent

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND APPARATUS FOR AUTOMATING TESTING OF JAVA BEANS

the specification of which

X is attached hereto.

___ was filed on _____ as
United States Application Number: _____
or PCT International Application Number _____
and was amended on _____.
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and that the invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to this application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

(Number)	(Country)	(Day/Month/Year Filed)	Priority Claimed Yes	No

I hereby claim the benefit under title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below:

(Application Number)	Filing Date

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Number)	Filing Date	(Status -- patented, pending, abandoned)

I hereby appoint the persons listed on Appendix A hereto (which is incorporated by reference and a part of this document) as my respective patent attorneys and patent agents, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to Kurt P. Leyendecker, BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, 12400 Wilshire Boulevard 7th Floor, Los Angeles, California 90025 and direct telephone calls to Kurt P. Leyendecker, (303) 740-1980.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: Gary Barnett

Inventor's Signature _____ Date 5/23/00

Residence San Jose, California (City, State) Citizenship USA (Country)

Post Office Address 1491 Cresthaven
San Jose, California 95135

Full Name of Second/Joint Inventor: Vaishali Ghanwat

Inventor's Signature Ghanwat Date 5/23/2000

Residence Milpitas, California (City, State) Citizenship India (Country)

Post Office Address: 1876 Grand Teton Drive
Milpitas, California 95035

APPENDIX A

William E. Alford, Reg. No. 37,764; Farzad E. Amini, Reg. No. P42,261; Aloysius T. C. AuYeung, Reg. No. 35,432; William Thomas Babbitt, Reg. No. 39,591; Carol F. Barry, Reg. No. 41,600; Jordan Michael Becker, Reg. No. 39,602; Bradley J. Bereznak, Reg. No. 33,474; Michael A. Bernadicou, Reg. No. 35,934; Roger W. Blakely, Jr., Reg. No. 25,831; Gregory D. Caldwell, Reg. No. 39,926; Ronald C. Card, Reg. No. P44,587; Thomas M. Coester, Reg. No. 39,637; Stephen M. De Klerk, under 37 C.F.R. § 10.9(b); Michael Anthony DeSanctis, Reg. No. 39,957; Daniel M. De Vos, Reg. No. 37,813; Robert Andrew Diehl, Reg. No. 40,992; Matthew C. Fagan, Reg. No. 37,542; Tarek N. Fahmi, Reg. No. 41,402; James Y. Go, Reg. No. 40,621; James A. Henry, Reg. No. 41,064; Willmore F. Holbrow III, Reg. No. P41,845; Sheryl Sue Holloway, Reg. No. 37,850; George W Hoover II, Reg. No. 32,992; Eric S. Hyman, Reg. No. 30,139; William W. Kidd, Reg. No. 31,772; Erica W. Kuo, Reg. No. 42,775; Kurt P. Leyendecker, Reg. No. 42,799; Gordon R. Lindeen, Reg. No. 33,192; Michael J. Mallie, Reg. No. 36,591; Andre L. Marais, under 37 C.F.R. § 10.9(b); Paul A. Mendonsa, Reg. No. 42,879; Darren J. Milliken, Reg. 42,004; Lisa A. Norris, Reg. No. P44,976; Chun M. Ng, Reg. No. 36,878; Thien T. Nguyen, Reg. No. 43,835; Thinh V. Nguyen, Reg. No. 42,034; Dennis A. Nicholls, Reg. No. 42,036; Kimberley G. Nobles, Reg. No. 38,255; Daniel E. Ovanezian, Reg. No. 41,236; Babak Redjaian, Reg. No. 42,096; William F. Ryann, Reg. 44,313; James H. Salter, Reg. No. 35,668; William W. Schaal, Reg. No. 39,018; James C. Scheller, Reg. No. 31,195; Jeffrey Sam Smith, Reg. No. 39,377; Maria McCormack Sobrino, Reg. No. 31,639; Stanley W. Sokoloff, Reg. No. 25,128; Judith A. Szepesi, Reg. No. 39,393; Vincent P. Tassinari, Reg. No. 42,179; Edwin H. Taylor, Reg. No. 25,129; John F. Travis, Reg. No. 43,203; George G. C. Tseng, Reg. No. 41,355; Joseph A. Twarowski, Reg. No. 42,191; Lester J. Vincent, Reg. No. 31,460; Glenn E. Von Tersch, Reg. No. 41,364; John Patrick Ward, Reg. No. 40,216; Charles T. J. Weigell, Reg. No. 43,398; James M. Wu, Reg. No. P45,241; Steven D. Yates, Reg. No. 42,242; Ben J. Yorks, Reg. No. 33,609; and Norman Zafman, Reg. No. 26,250; my patent attorneys, and Andrew C. Chen, Reg. No. 43,544; Justin M. Dillon, Reg. No. 42,486; Paramita Ghosh, Reg. No. 42,806; and Sang Hui Kim, Reg. No. 40,450; my patent agents, of BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, with offices located at 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025, telephone (310) 207-3800, and James R. Thein, Reg. No. 31,710, my patent attorney; and Mark J. Meltzer, Reg. No. 28,739 of ASPECT COMMUNICATIONS, with offices located at 1310 Ridder Park, San Jose, CA 95131.

APPENDIX B

Title 37, Code of Federal Regulations, Section 1.56
Duty to Disclose Information Material to Patentability

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclosure information exists with respect to each pending claim until the claim is cancelled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is cancelled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclosure all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) Prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) The closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made or record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application;

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.